REMARKS

STATUS OF THE CLAIMS

Claims 1-16 were pending and stand rejected.

By this Amendment, claims 1-16 are amended and new claims 17-19 are added. Therefore, claims 1-19 are now presented for consideration.

No new matter is presented in the foregoing specification, claim and drawing amendments and new claims, and accordingly, approval and entry of same are submitted to be proper and are respectfully solicited.

DRAWING OBJECTIONS

In the Office Action, at page 2, item 1, the drawings are objected to for various reasons provided therein.

It is submitted that the objection to the drawings is now overcome because: a) one of the reference characters "502" has been amended to --501--; b) the specification has been amended to conform with FIG. 29 with respect to reference character "il"; and c) the specification has been amended to conform with FIG. 7 with respect to reference character "a1."

Moreover, Applicants traverse the objection with respect to "b2" of FIG. 9 not being mentioned in the description, since "b2" of FIG. 9 is mentioned in the specification at page 35, line 12.

Reconsideration is respectfully requested.

SPECIFICATION OBJECTIONS

In the Office Action, at page 2, item 2, the disclosure is objected to because of informalities therein and, furthermore, the Examiner requested a copy of the convolution algorithm described at page 30, lines 16-18 and page 36, lines 19-22 and page 38, lines 23-24 of the specification.

Reconsideration is respectfully requested.

The specification is amended to overcome the objections thereto.

Moreover, a copy of the convolution algorithm, as requested by the Examiner, is enclosed, as Attachment 2.

CLAIM REJECTIONS UNDER 35 U.S.C. §112, SECOND PARAGRAPH

In the Office Action, at pages 3-4, item 4, claims 1-16 are rejected under 35 U.S.C. §112, second paragraph, as being indefinite.

Claims 1-16 are amended to overcome the rejection thereto.

Reconsideration is respectfully requested.

CLAIM REJECTION UNDER 35 U.S.C. §102 (e)

Claims 1-3 and 5-16 are rejected under 35 U.S.C. 102(e) as being anticipated by Ogura et al. (U.S. Patent No. 6,502,984 B2) hereafter referred to as Ogura 984.

Claim 1 is directed to an image processing apparatus, and recites "an image processing condition storing section to store an image processing condition when the radiation image is subjected to the image processing in accordance with a type of the photography device and a part of the target when the radiation image is obtained ... an image processing section ... to subject the radiation image ... to the image processing in accordance with the image processing condition." The image processing condition is stored when the radiation image is subjected to the image processing. That is, the radiation image or radiation image data exists and is subject to the image processing, the image processing being in accordance with the image processing condition. Furthermore, since the image processing apparatus of the present invention is provided with an image processing condition storing section, an radiation image is optimally processed based on the stored image processing condition, irrespective of the type of photographic device that took the radiation image and the part of the target as the subject of radiation image.

Ogura 984 Reference

Ogura 984 discloses "to provide an X-ray photographing apparatus which determines the X-ray irradiation time by measuring the body thickness of an object before X-ray irradiation, and

controls the moving grid on the basis of the determined X-ray irradiation time information to remove the influences of shadow images of lead foils formed in an X-ray image." (See Ogura 984 at column 5, lines 53-59.) Thus, in the Ogura 984 apparatus, image processing is performed prior to X-ray irradiation such that an X-ray image is not subjected to the image processing in accordance with the image processing condition (see claim 1). This means that the Ogura 984 image processing is different from "the image processing," as recited in claim 1.

Furthermore, Ogura 984 is silent with regard to "an image processing condition storing section to store an image processing condition" associated with a type of the photography device and a part of the target," because in the Ogura 984 apparatus the image processing condition is measured (e.g., by measuring the body thickness). Thus, an image processing condition storing section is not disclosed or suggested by Ogura 984.

Accordingly, claim 1 patentably distinguish over the cited art and is submitted to be allowable.

Claims 2, 15 and 16

Claims 2, 15 and 16, for reason similar to those of claim 1 are submitted to also be allowable.

Claims 3-14

Claims 3-14, which depend from claim 2, are also submitted to be allowable for at least the same reasons as claim 2, as well as for the additional recitations therein.

CLAIM REJECTION UNDER 35 U.S.C. §103(a)

Claim 4 is rejected under 35 U.S.C. 103(a) as being unpatentable over the combination of Ogura 984 and Ogura (U.S. Patent No. 6,314,198 B1) hereafter referred to as Ogura 198.

It is submitted that the additional reference of Ogura 198 does not overcome the deficiencies of Ogura 984 because Ogura 198 does not disclose or suggest, in the portions cited by the Examiner, anything related to the recitation in claim 1 of "an image processing condition storing section to store an image processing condition when the radiation image is subjected to the image processing in accordance with a type of the photography device and a part of the target when the radiation image is obtained."

NEW CLAIMS 17-19

New claims 17-19 are provide to afford a varying scope of protection.

New claim 17 recites "an image processing condition table, having indices of parts of a body and types of photography devices used to image medical images, to store and output an image processing condition; and an image processor to enhance the medical image based on the image processing condition stored in the image processing condition table," and is submitted to be allowable.

New claims 18 and 19 recite storing an image processing condition in a table having indices of parts of a body and types of a photography devices used to image medical images; and subjecting the medical image to processing based on the stored image processing condition," and are submitted to be allowable.

Entry and consideration of this claim is respectfully requested.

CONCLUSION

There being no further outstanding objections or rejections, it is submitted that the application is in condition for allowance. An early action to that effect is courteously solicited.

Finally, if there are any formal matters remaining after this response, the Examiner is requested to telephone the undersigned to attend to these matters.

If there are any additional fees associated with filing of this Amendment, please charge the same to our Deposit Account No. 19-3935.Respectfully submitted,

STAAS & HALSEY LLP

Date: 4/20/04

By: Eric Berkowitz
Eric Berkowitz
Registration No. 44,030

1201 New York Avenue, NW, Suite 700
Washington, D.C. 20005
Telephone: (202) 434-1500
Facsimile: (202) 434-1501

CERTIFICATE UNDER 37 CFR 1.8(a)
I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, PO Box 1450, Alexandria, VA 22313-1450
on 4/20/04 ,20 ____
STAAS & HALSEY
By: Eric Berkowitz
Date: 4/20/04

ATTACHMENT 1
TO THE AMENDMENT
FILED APRIL 20, 2004
IN SERIAL NO. 09/714,292
DOCKET NO. 1080.1084

ATTACHMENT 2
TO THE AMENDMENT
FILED APRIL 20, 2004
IN SERIAL NO. 09/714,292
DOCKET NO. 1080.1084

FILTERING OPERATOR CONVOLVE, 2D

Description:

Performs two-dimensional convolution with a general impulse response function array for several image boundary conditions.

Name:

convolve_2d

Parameters:

SRC	source image	ID	in
DST	destination image	ID	out
SOB	impulse response array object	ID	in
convolution	convolution option	SP	in
	upper left corner justified		
	enclosed array		
	key pixel, zero exterior		
	key pixel, reflected exterior		

Objects:

SRC structure:	any image
DST structure:	same as SRC structure
SOB structure:	NBHOOD_ARRAY
SRC data type:	ND, SD, RD
DST data type:	same as SRC data type
SOB data type:	SD, RD
SOB dimension:	2D

Element functionality:

Index assignment:	yes
Match point:	yes
ROI control:	yes
ROI processing:	no
Source promotion:	no
Heterogeneous bands:	no
Chainable element:	yes
Asynchronous:	yes

Remarks:

1. The PIKS data object repository contains a set of fixed size impulse response arrays.
2. Variable size impulse response arrays can be generated by PIKS tools.
3. One-dimensional convolution can be achieved with this operator by specifying an impulse response array with only a single row or column.
4. The impulse response array data type shall be SD if the source image data type is ND or SD.
5. The impulse response array data type shall be RD if the source image data type is RD.

Nomenclature:

SRC(x, y, z, t, b)	$X_S x Y_S x Z_S x T_S x B_S$ source image
DST(x, y, z, t, b)	$X_D x Y_D x Z_D x T_D x B_D$ destination image
S(j, k, l, m, n)	$J_S x K_S x L_S x M_S x N_S$ operator input image
D(j, k, l, m, n)	$J_D x K_D x L_D x M_D x N_D$ operator output image
H(j, k)	$J_H x K_H$ impulse response function array
(j_k, k_k)	key pixel coordinate

The key pixel coordinate is referenced to the origin of the impulse response function array. It is not restricted to be within the bounds of the impulse response function array, i.e., j_k can be negative or larger than J_H.

Definition:

Upper left corner justified mode:

The 2D impulse response function array is rotated 180 degrees and translated with respect to the j-k pixel plane of the source image. A destination pixel is computed for all spatial intersections of the array and the source image. The destination image j-k pixel plane is indexed from its upper left corner. The destination image j-k plane is larger than the source image j-k plane.

For all l, m, n, and for

$$0 \le j \le J_D - 1$$

$$0 \le k \le k_D - 1$$

$$D(j,k,l,m,n) = \sum_{j'} \sum_{k'} S(j',k',l,m,n)\, H(j-j', k-k')$$

The summation limits are

$$MAX\{0, j - J_H + 1\} \le j' \le MIN\{J_S - 1, j\}$$

$$MAX\{0, k - K_H + 1\} \le k' \le MIN\{K_S - 1, k\}$$

Algorithm described at page 36, lines 19-22

The pixel plane sizes are

$$J_D = J_S + J_H - 1$$

$$K_D = K_S + K_H - 1$$

$$L_D = L_S$$

$$M_D = M_S$$

$$N_D = N_S$$

Enclosed impulse response array mode:

The 2D impulse response function array is rotated 180 degrees and translated with respect to the j-k pixel plane of the source image. A destination pixel is computed whenever all elements of the array lie within the bounds of the source image; all other destination pixels are set to zero. The destination image size is the same as the source image size.

For all l, m, n, and for

$$j_c \le j \le J - 1 - j_c \qquad \text{if } J_H \text{ is odd}$$

$$j_c - 1 \le j \le J - 1 - j_c \qquad \text{if } J_H \text{ is even}$$

$$k_c \le k \le K - 1 - k_c \qquad \text{if } K_H \text{ is odd}$$

$$k_c - 1 \le k \le K - 1 - k_c \qquad \text{if } K_H \text{ is even}$$

$$D(j, k, l, m, n) = \sum_{j'} \sum_{k'} S(j', k', l, m, n)\, H(j - j' + j_c, k - k' + k_c)$$

For all l, m, n, and for

$$j < j_c \quad \text{or} \quad j > J - 1 - j_c \qquad \text{if } J_H \text{ is odd}$$

$$j < j_c - 1 \quad \text{or} \quad j > J - 1 - j_c \qquad \text{if } J_H \text{ is even}$$

$$k < k_c \quad \text{or} \quad k > K - 1 - k_c \qquad \text{if } K_H \text{ is odd}$$

$$k < k_c - 1 \quad \text{or} \quad k > K - 1 - k_c \qquad \text{if } K_H \text{ is even}$$

$$D(j, k, l, m, n) = 0$$

where

$$j_c = \frac{J_H - 1}{2} \qquad \text{if } J_H \text{ is odd}$$

$$j_c = \frac{J_H}{2} \qquad \text{if } J_H \text{ is even}$$

$$k_c = \frac{K_H - 1}{2} \qquad \text{if } K_H \text{ is odd}$$

$$k_c = \frac{K_H}{2} \qquad \text{if } K_H \text{ is even}$$

The summation limits are

$$MAX\{0, j-j_c\} \le j' \le MIN\{J-1, j+j_c\} \qquad \text{if } J_H \text{ is odd}$$

$$MAX\{0, j-j_c+1\} \le j' \le MIN\{J-1, j+j_c\} \qquad \text{if } J_H \text{ is even}$$

$$MAX\{0, k-k_c\} \le k' \le MIN\{K-1, k+k_c\} \qquad \text{if } K_H \text{ is odd}$$

$$MAX\{0, k-k_c+1\} \le k' \le MIN\{K-1, k+k_c\} \qquad \text{if } K_H \text{ is even}$$

The pixel plane sizes are

$$J_D = J_S = J$$
$$K_D = K_S = K$$
$$L_D = L_S = L$$
$$M_D = M_S = M$$
$$N_D = N_S = N$$

Key pixel, zero exterior mode:

The 2D impulse response function array is rotated 180 degrees and translated with respect to the j-k pixel plane of the source image. A destination pixel is computed whenever the key pixel of the array is over a source pixel. The destination image size is the same as the source image size. Source pixels outside the j-k plane boundary are assumed to be zero valued.

For all l, m, n, and for

$$MAX\{0, -j_k\} \le j \le MIN\{J-1, J-1+j_k\}$$
$$MAX\{0, -k_k\} \le k \le MIN\{K-1, K-1+k_k\}$$

$$D(j, k, l, m, n) = \sum_{j'} \sum_{k'} S(j', k', l, m, n)\, H(j - j' + j_k, k - k' + k_k)$$

For all l, m, n, and for

$$j < MAX\{0, -j_k\} \quad \text{or} \quad j > MIN\{J-1, J-1+j_k\}$$
$$k < MAX\{0, -k_k\} \quad \text{or} \quad k > MIN\{K-1, K-1+k_k\}$$

$$D(j, k, l, m, n) = 0$$

The summation limits are

$$MAX\{0, j-J_H+j_k+1\} \le j' \le MIN\{J-1, j+j_k\}$$

$$MAX\{0, k-K_H+k_k+1\} \le k' \le MIN\{K-1, k+k_k\}$$

The pixel plane sizes are

$$J_D = J_S = J$$

$$K_D = K_S = K$$

$$L_D = L_S = L$$

$$M_D = M_S = M$$

$$N_D = N_S = N$$

Algorithm described at page 30, lines 16-18

Key pixel, reflected exterior mode:

The 2D impulse response function array is rotated 180 degrees and translated with respect to the j-k pixel plane of the source image. A destination pixel is computed whenever the key pixel of the impulse response array is over a source pixel. The destination image size is the same as the source image size. Source pixels outside the j-k pixel plane boundary are assumed to be reflections of source pixels about each boundary, i.e., the source image is assumed to be reflected about its left, right, top, and bottom sides.

For all l, m, n, and for

$$0 \le j \le J-1$$

$$0 \le k \le K-1$$

$$D(j, k, l, m, n) = \sum_{j'} \sum_{k'} S(j'', k'', l, m, n)\, H(j-j''+j_{k'}, k-k''+k_{k'})$$

where

$j'' = 1-j'$	$j' < 0$	$k'' = 1-k'$	$k' < 0$
$j'' = j'$	$0 \le j' \le J-1$	$k'' = k'$	$0 \le k' \le K-1$
$j'' = 2(J-1)-j'$	$j' > J-1$	$k'' = 2(K-1)-k'$	$k' > K-1$

The summation limits are

$$MAX\{0, j-J_H+j_k+1\} \le j' \le MIN\{J-1, j+j_k\}$$

$$MAX\{0, k-K_H+k_k+1\} \le k' \le MIN\{K-1, k+k_k\}$$

The pixel plane sizes are

$$J_D = J_S = J$$

$$K_D = K_S = K$$

$$L_D = L_S = L$$

$$M_D = M_S = M$$

$$N_D = N_S = N$$

Profile:

Foundation

Error codes:

001
003
011
012
013
014
041
045
052
061

題名「ＣＱ出版社　改訂Ｃによる科学技術計算」
著作「小池　汎一」
出版社「ＣＱ出版社」

↑ Algorithm described at page 38, lines 23-24
(English attached hereto)

〈表3.1.2〉 ベース・スプラインのはじめの4点のx, x', x''の値

n	0	1	2	3
x	x_0	$\frac{1}{12}(2x_3+7x_2+3x_1)$	$\frac{1}{6}(x_4+4x_3+x_2)$	$\frac{1}{6}(x_5+4x_4+x_3)$
x'	$3(x_1-x_0)$	$\frac{1}{4}(2x_3+x_2-3x_1)$	$\frac{1}{2}(x_4-x_2)$	$\frac{1}{2}(x_5-x_3)$
x''	$3(x_2-3x_1+2x_0)$	$\frac{1}{2}(2x_3-5x_2+3x_1)$	$x_4-2x_3+x_2$	$x_5-2x_4+x_3$

雲形定規的スプライン

ふつう雲形定規を使って点と点の間を結ぶ場合，各点でなめらかになるように結びます．“なめらかに”ということは，前後の区間に対してデータ点における1次微係数が同じであることを意味します．当然，データ点を通ります．このような補間の方法を「雲形定規的補間[1]」と名づけ，ここでは“雲形定規的スプライン”と呼ぶことにします．

連続する4点のデータを$x_{i-1}, x_i, x_{i+1}, x_{i+2}$とします($y$軸方向のデータは省略します)．正規化されたパラメータを$u(0\leqq u\leqq 1)$として，3次式を考えます．

$$x_i(u) = b_0^{(i)}+b_1^{(i)}u+b_2^{(i)}u^2+b_3^{(i)}u^3 \quad (i=1,2,\cdots,N-1, 0\leqq u\leqq 1) \tag{15}$$

$x_i(u)$は補間によって得られたx軸方向の値です．y軸方向についても同様に得られます．まず，係数$b_0^{(i)}, b_1^{(i)}, b_2^{(i)}, b_3^{(i)}$を求めるために，条件を与えます．

$$x_i(0) = b_0^{(i)} = x_i \tag{16.a}$$

$$x_i(1) = b_0^{(i)}+b_1^{(i)}+b_2^{(i)}+b_3^{(i)} = x_{i+1} \tag{16.b}$$

$$x'_i(0) = b_1^{(i)} = \frac{1}{2}(x_{i+1}-x_{i-1}) \tag{16.c}$$

$$x'_i(1) = b_1^{(i)}+2b_2^{(i)}+3b_3^{(i)} = \frac{1}{2}(x_{i+2}-x_i) \tag{16.d}$$

ここにx_iとx_{i+1}における1次微係数は，等間隔な3点に2次式を当てはめた場合のそれを与えています．式(16)より各係数を求め，xについて整理すると

$$x_i(u) = \frac{1}{2}u^2(u-1)x_{i+2}+\frac{1}{2}u(1+4u-3u^2)x_{i+1}+\frac{1}{2}(u-1)(3u^2-2u-2)x_i$$
$$-\frac{1}{2}u^2(u-1)x_{i-1} \quad (i=1,2,\cdots,N-1,\ 0\leqq u\leqq 1) \tag{17}$$

を得ます．

はじめと終わりの区間に対しては，式(17)は利用できません．$u=-1$および$u=2$

は，点 x_{i-1} と x_{i+2} を与えないからです(混合関数は形がラグランジェの補間と同一になるので，具合よくそれらの点を通りましたが…)．そこで両端に対しては，別の条件を与えることにします．はじめの区間を与える補間を $x_0(u)$ として，$-1 \leqq u \leqq 0$ で補間するとします．

$$x_0(u) = b_0^{(0)} + b_1^{(0)}u + b_2^{(0)}u^2 + b_3^{(0)}u^3 \quad (-1 \leqq u \leqq 0) \tag{18}$$

条件としては始点 x_0 と点 x_1 を通ること，点 x_1 の1次微係数が隣りの区間のそれと一致することを与えます．これだけでは条件が足りず係数を定められないので，点 x_2 においても微係数が一致するものとします．条件を書き下すと，

$$x_0(-1) = b_0^{(0)} - b_1^{(0)} + b_2^{(0)} - b_3^{(0)} = x_0 \tag{19.a}$$

$$x_0(0) = b_0^{(0)} = x_1 \tag{19.b}$$

$$x_0'(0) = b_1^{(0)} = \frac{1}{2}(x_2 - x_0) \tag{19.c}$$

$$x_0'(1) = b_1^{(0)} + 2b_2^{(0)} + 3b_3^{(0)} = \frac{1}{2}(x_3 - x_1) \tag{19.d}$$

となります．これを解いて x について整理すると，

$$x_0(u) = \frac{1}{10}u^2(1+u)x_3 + \frac{1}{10}u(u+1)(5-3u)x_2 + \frac{1}{10}(u+1)(3u^2-10u+10)x_1 - \frac{1}{10}u(u^2-4u+5)x_0 \quad (-1 \leqq u \leqq 0) \tag{20}$$

を得ます．

同様にして終わりの区間についても，

$$x_N(u) = \frac{1}{10}(u-1)(u^2+2u+2)x_N - \frac{1}{10}(u-2)(3u^2+4u+3)x_{N-1} + \frac{1}{10}(u-1)(u-2)(3u+2)x_{N-2} - \frac{1}{10}(u-1)^2(u-2)x_{N-3} \quad (1 \leqq u \leqq 2) \tag{21}$$

を得ます．

このスプラインによる補間は，混合関数と似た結果を与えます．そしてプログラムも同じような形をとります．式の形が簡単なぶんだけ混合関数が優れているように思えますが，混合関数の性質が2次微係数が一致することと直観性を欠くのに対して，このスプラインでは，データ点における1次微係数が一致するので，常識的といえます．

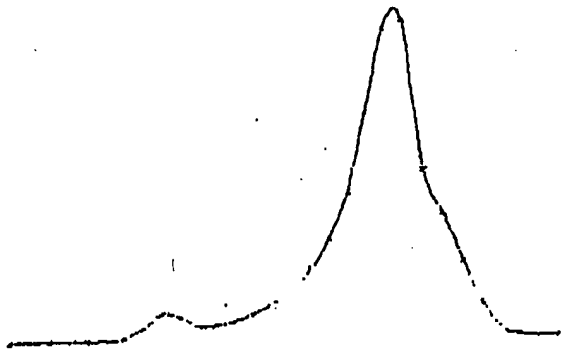
●雲形定規的スプラインのプログラミング

List 3.1.5に示します．同プログラムは関数 set_blend() の内容を除いて，混合スプラインと同じになります．特別な処置を必要とするのは，はじめと終わりの区間のみで

〈List 3.1.5〉雲形定規的スプライン・プログラム（次ページへつづく）

```
#include  <stdio.h>
#include  <stdlib.h>
#include  <conio.h>
#include  <graphics.h>
#include  "spline.h"

#define  SPLIT    8
#define  X_RANGE  0.8
#define  Y_RANGE  0.8

double  x1[100], y1[100], wx[SPLIT+1], wy[SPLIT+1], d;
double  b[SPLIT+1][4], first_b[SPLIT+1][4], last_b[SPLIT+1][4];
void set_blend(double (*b)[4], double (*f)[4], double (*l)[4], int n);
void mk_blend(double* x, double* y, double* wx, double* wy,
              double (*b)[4], int n);

void main()
{
    double  kx, ky, xmax, ymax, xmin, ymin;
    int     x0, y0, y_height, x_width, n, k, cl;
    char    *s=("data1");

    n = rd_data(s, x1, y1, &xmin, &xmax, &ymin, &ymax);
                                /* read data from a file 's' */
    opengraph("a:\\tc\\bgi", &x_width, &y_height);

    cl = getmaxcolor();
    kx = x_width * X_RANGE / (xmax - xmin);
    ky = y_height * Y_RANGE / (ymax - ymin);
    x0 = x_width * (1-X_RANGE) / 2 - xmin * kx;
    y0 = y_height * (1-Y_RANGE) / 2 - ymin * ky;

    plot_data(x1, y1, n, cl, x0, y0, kx, ky, y_height);

/*  draft spline sample program for open curve */
    set_blend(b, first_b, last_b, SPLIT);   /* set coeffcients */
    k = 0;                                  /* data counter */
    mk_blend(&x1[k], &y1[k], wx, wy, first_b, SPLIT);
                                            /* first section */
    mk_curve(wx, wy, SPLIT, cl, x0, y0, kx, ky, y_height);
    while( k < n-3 ) {
        mk_blend(&x1[k], &y1[k], wx, wy, b, SPLIT);
        mk_curve(wx, wy, SPLIT, cl, x0, y0, kx, ky, y_height);
        k++;
    }
    k--;
    mk_blend(&x1[k], &y1[k], wx, wy, last_b, SPLIT);
    mk_curve(wx, wy, SPLIT, cl, x0, y0, kx, ky, y_height);
    getch();
    closegraph();
}
```

```
void set_blend(double (*b)[4], double (*f)[4], double (*l)[4], int n)
{
    double  u;
    int     i;
    for( i = 0; i <= n; i++ ) {
        u = (double)i / (double)n;
        b[i][0] = -u * (u - 1.0) * (u - 1.0) / 2.0;
        b[i][1] = (u - 1.0) * (3.0 * u * u - 2.0 * u - 2.0) / 2.0;
        b[i][2] = u * (1.0 + 4.0 * u - 3.0 * u * u) / 2.0;
        b[i][3] = u * u * (u - 1.0) / 2.0;

        u = (double)i / (double)n - 1.0;
        f[i][0] = -u * (u * u - 4.0 * u + 5.0) / 10.0;
        f[i][1] = (u + 1.0) * (3.0 * u * u - 10.0 * u + 10.0) / 10.0;
        f[i][2] = u * (u + 1.0) * (5.0 - 3.0 * u) / 10.0;
        f[i][3] = u * u * (u + 1.0) / 10.0;

        u = (double)i / (double)n + 1.0;
        l[i][0] = -(u - 1.0) * (u - 1.0) * (u - 2.0) / 10.0;
        l[i][1] = (u - 1.0) * (u - 2.0) * (3.0 * u + 2.0) / 10.0;
        l[i][2] = -(u - 2.0) * (3.0 * u * u + 4.0 * u + 3.0) / 10.0;
        l[i][3] = (u - 1.0) * (u * u + 2.0 * u + 2.0) / 10.0;
    }
}
```



●実行例●

す．関数mk_blend()は同一なのでリストからは省略されています．例として用いたデータでは，混合スプラインの場合とほとんど同一の曲線が得られます．

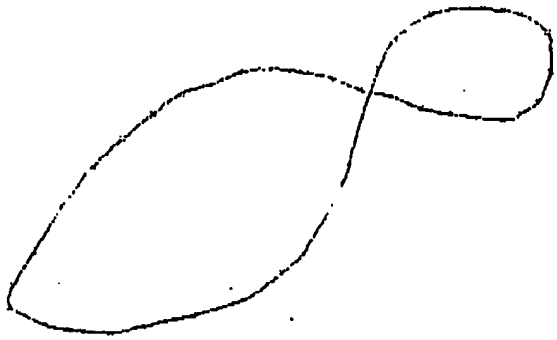
●閉じた曲線のためのプログラム

プログラムをList 3.1.6に示します．関数set_blend()が，前出のList 3.1.5のものである点を除けば，混合関数のものと同じです．

<List 3.1.6> 閉じた曲線のための雲形定規的スプライン・プログラム

```
/*  draft spline sample program for closed curve */
    set_blend(b, first_b, last_b, SPLIT);    /* set coeffcients */
    k = 0;                                   /* data counter */
    while( k < n-3 ) {
        mk_blend(&x1[k], &y1[k], wx, wy, b, SPLIT);
        mk_curve(wx, wy, SPLIT, c1, x0, y0, kx, ky, y_height);
        k++;
    }
    getch();
    closegraph();
}
```

●実行例●



3次スプライン

これまでに述べた混合スプライン，ベース・スプライン，雲形定規的スプラインは，厳密な意味ではスプライン関数とは呼ばれていません．しかし，式が簡単で計算も容易であることと，つぎに述べる3次スプラインと比べても，補間性能(ふつうの実験データ――極端なピークや谷を含まない)ではあまり差はありません．そのため，ここでは一括して“スプライン”の名称を与えています．

スプラインとは文献(5)によれば，“しなやかな弾性体でできた帯”のことです．その帯を用いてなめらかな曲線を得るには，ところどころを鉛などでできた重りで固定して他端を自由にさせます．このようにすると，スプラインは弾性体の性質にしたがう曲線を描くので，それを用いるとなめらかな曲線が得られます．数学的には各区間において $k-1$ 次の微係数が一致する，k 次の多項式のことをスプラインといいます．多項式の次数は高ければよいというものでもなく，ふつうは2次とか3次とかを用います．ここでは3次とします．

3次スプラインは各区間において2次までの微係数が一致し，かつデータ点を通ります．このような3次式の係数は，前4節までで用いたように連続する4点のデータのみからは決定できません．全データを用い，あとで示すようにさらに不足の条件を与えてはじ

Title : "SCIENTIFIC CALCULATION USING C"
Author : Shinichi Koike
Publisher : CQ Publishing Co., Ltd.

Chapter 3 APPLICATION PROGRAMMING
3.1 Interpolation and Spline

EXTRACTION

<Table 3.1.2> Values of x, x', and x'' of starting four points of base spline

n	0	1	2	3
x	x_0	$\frac{1}{12}(2x_3+7x_2+3x_1)$	$\frac{1}{6}(x_4+4x_3+x_2)$	$\frac{1}{6}(x_5+4x_4+x_3)$
x'	$3(x_1-x_0)$	$\frac{1}{4}(2x_3+x_2-3x_1)$	$\frac{1}{2}(x_4-x_2)$	$\frac{1}{2}(x_5-x_3)$
x''	$3(x_2-3x_1+2x_0)$	$\frac{1}{2}(2x_3-5x_2+3x_1)$	$x_4-2x_3+x_2$	$x_5-2x_4+x_3$

Curve rule-like spline

When one dot and another are interconnected by using a curved rule, they are interconnected in such a manner that an interconnecting line may be smooth at each of these points. To be "smooth" means that a linear differential coefficient at a data point is the same in the immediately preceding interval and in the immediately following interval. Of course, the line passes through the data point. Such a method of interpolation is referred to as "curved rule-like interpolation[1]" and as "curved rule-like spline" here.

Assume four consecutive data pieces to be X_{i-1}, X_i, X_{i+1}, and X_{i+2} (where y-axial data is omitted). Assuming a normalized parameter to be $u(0 \leq u \leq 1)$, the following cubic equation is considered:

$$x_i(u) = b_0^{(i)} + b_1^{(i)}u + b_2^{(i)}u^2 + b_3^{(i)}u^3 \quad (i=1,2,\cdots,N-1, 0 \leqq u \leqq 1) \tag{15}$$

where $X_i(u)$ is an x-axial value obtained by interpolation. A y-axial value can also be obtained similarly. First, to obtain coefficients $b_0^{(i)}$, $b_1^{(i)}$, $b_2^{(i)}$, and $b_3^{(i)}$, conditions are given as follows:

$$x_i(0) = b_0^{(i)} = x_i \tag{16.a}$$

$$x_i(1) = b_0^{(i)} + b_1^{(i)} + b_2^{(i)} + b_3^{(i)} = x_{i+1} \tag{16.b}$$

$$x'_i(0) = b_1^{(i)} = \frac{1}{2}(x_{i+1} - x_{i-1}) \tag{16.c}$$

$$x'_i(1) = b_1^{(i)} + 2b_2^{(i)} + 3b_3^{(i)} = \frac{1}{2}(x_{i+2} - x_i) \tag{16.d}$$

where linear differential coefficients at the respect points of X_i and X_{i+1} provide coefficients in a case where a quadratic equation is applied to three equally spaced points. By obtaining each of the coefficients from Equation (16) and organizing it, the following equation is obtained:

$$x_i(u) = \frac{1}{2}u^2(u-1)x_{i+2} + \frac{1}{2}u(1+4u-3u^2)x_{i+1} + \frac{1}{2}(u-1)(3u^2-2u-2)x_i - \frac{1}{2}u^2(u-1)x_{i-1} \quad (i=1,2,\cdots,N-1,\ 0 \leqq u \leqq 1) \tag{17}$$

Equation (17) cannot be used for a first interval and a last interval. This is because $u=-1$ and $u=2$ do not give points X_{i-1} and X_{i+2} respectively (although these points have been passed though because a mixed function gives the same shape as that of Lagrange's interpolation). Therefore, for both ends, different conditions are given. That is, assuming an interpolation that gives a first interval to be $X_0(u)$ and performed in a range of $-1 \leqq u \leqq 0$, the following equation is given:

$$x_0(u) = b_0^{(0)} + b_1^{(0)}u + b_2^{(0)}u^2 + b_3^{(0)}u^3 \quad (-1 \leqq u \leqq 0) \tag{18}$$

The conditions are such that a starting point X_0 and a point X_1 should be passed through and a linear differential coefficient at the point X_1 should agree with that of an adjacent interval. Since these conditions alone are not enough to determine a coefficient, another condition is set such that the differential coefficients should agree also at a point X_2. The conditions are written as follows:

$$x_0(-1) = b_0^{(0)} - b_1^{(0)} + b_2^{(0)} - b_3^{(0)} = x_0 \tag{19.a}$$

$$x_0(0) = b_0^{(0)} = x_1 \tag{19.b}$$

$$x_0'(0) = b_1^{(0)} = \frac{1}{2}(x_2 - x_0) \tag{19.c}$$

$$x_0'(1) = b_1^{(0)} + 2b_2^{(0)} + 3b_3^{(0)} = \frac{1}{2}(x_3 - x_1) \tag{19.d}$$

These are solved and organized in terms of x as follows:

$$x_0(u) = \frac{1}{10}u^2(1+u)x_3+\frac{1}{10}u(u+1)(5-3u)x_2+\frac{1}{10}(u+1)(3u^2-10u+10)x_1$$
$$-\frac{1}{10}u(u^2-4u+5)x_0 \qquad (-1\leqq u\leqq 0) \tag{20}$$

Similarly, in an ending interval, the following is obtained:

$$x_N(u) = \frac{1}{10}(u-1)(u^2+2u+2)x_N-\frac{1}{10}(u-2)(3u^2+4u+3)x_{N-1}$$
$$+\frac{1}{10}(u-1)(u-2)(3u+2)x_{N-2}-\frac{1}{10}(u-1)^2(u-2)x_{N-3} \tag{21}$$
$$(1\leqq u\leqq 2)$$

This interpolation by use of a spline gives results similar to those by a mixed function. A relevant program also takes on a similar form. Although the mixed function may seem to be better just because the equation is simpler in form, it may be common knowledge that the mixed function has such properties that the quadratic differential coefficients agree and it lacks intuitiveness, whereas by this spline the linear differential coefficients agree at data points.

- **Programming of curved rule-like spline**

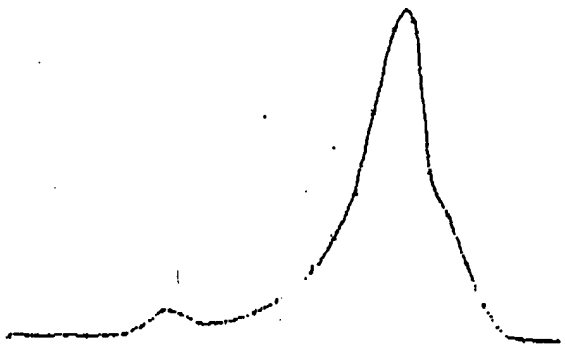
Programming of curved rule-like spline is indicated in List 3.1.5. This programming is the same as that of a mixed spline except contents of a function set_blend(). Special processing is required only in a starting interval and an ending interval. The function mk_blend() is the same and so omitted in the list. Based on data which has been used as an example, almost the same curve as that by the mixed spline is obtained.

<List 3.1.5> Curved rule-like spline program

(continued to next page)

```
#include  <stdio.h>
#include  <stdlib.h>
#include  <conio.h>
#include  <graphics.h>
#include  "spline.h"

#define  SPLIT    8
#define  X_RANGE  0.8
#define  Y_RANGE  0.8

double  x1[100], y1[100], wx[SPLIT+1], wy[SPLIT+1], d;
double  b[SPLIT+1][4], first_b[SPLIT+1][4], last_b[SPLIT+1][4];
void set_blend(double (*b)[4], double (*f)[4], double (*l)[4], int n);
void mk_blend(double* x, double* y, double* wx, double* wy,
              double (*b)[4], int n);

void main()
{
    double  kx, ky, xmax, ymax, xmin, ymin;
    int     x0, y0, y_height, x_width, n, k, cl;
    char    *s={"data1"};

    n = rd_data(s, x1, y1, &xmin, &xmax, &ymin, &ymax);
                                  /* read data from a file 's' */
    opengraph("a:\\tc\\bgi", &x_width, &y_height);

    cl = getmaxcolor();
    kx = x_width * X_RANGE / (xmax - xmin);
    ky = y_height * Y_RANGE / (ymax - ymin);
    x0 = x_width * (1-X_RANGE) / 2 - xmin * kx;
    y0 = y_height * (1-Y_RANGE) / 2 - ymin * ky;

    plot_data(x1, y1, n, cl, x0, y0, kx, ky, y_height);

/*  draft spline sample program for open curve */
    set_blend(b, first_b, last_b, SPLIT);   /* set coeffcients */
    k = 0;                                  /* data counter */
    mk_blend(&x1[k], &y1[k], wx, wy, first_b, SPLIT);
                                            /* first section */
    mk_curve(wx, wy, SPLIT, cl, x0, y0, kx, ky, y_height);
    while( k < n-3 ) {
        mk_blend(&x1[k], &y1[k], wx, wy, b, SPLIT);
        mk_curve(wx, wy, SPLIT, cl, x0, y0, kx, ky, y_height);
        k++;
    }
    k--;
    mk_blend(&x1[k], &y1[k], wx, wy, last_b, SPLIT);
    mk_curve(wx, wy, SPLIT, cl, x0, y0, kx, ky, y_height);
    getch();
    closegraph();
}
```

```
void set_blend(double (*b)[4], double (*f)[4], double (*l)[4], int n)
{
    double  u;
    int     i;
    for( i = 0; i <= n; i++ ) {
        u = (double)i / (double)n;
        b[i][0] = -u * (u - 1.0) * (u - 1.0) / 2.0;
        b[i][1] = (u - 1.0) * (3.0 * u * u - 2.0 * u - 2.0) / 2.0;
        b[i][2] = u * (1.0 + 4.0 * u - 3.0 * u * u) / 2.0;
        b[i][3] = u * u * (u - 1.0) / 2.0;

        u = (double)i / (double)n - 1.0;
        f[i][0] = -u * (u * u - 4.0 * u + 5.0) / 10.0;
        f[i][1] = (u + 1.0) * (3.0 * u * u - 10.0 * u + 10.0) / 10.0;
        f[i][2] = u * (u + 1.0) * (5.0 - 3.0 * u) / 10.0;
        f[i][3] = u * u * (u + 1.0) / 10.0;

        u = (double)i / (double)n + 1.0;
        l[i][0] = -(u - 1.0) * (u - 1.0) * (u - 2.0) / 10.0;
        l[i][1] = (u - 1.0) * (u - 2.0) * (3.0 * u + 2.0) / 10.0;
        l[i][2] = -(u - 2.0) * (3.0 * u * u + 4.0 * u + 3.0) / 10.0;
        l[i][3] = (u - 1.0) * (u * u + 2.0 * u + 2.0) / 10.0;
    }
}
```



•Execution example•

- **Program for closed curve**

A program is indicated in List 3.1.6. It is the same as that of the mixed function except that the function set_blend() is such as given in List 3.1.5.

<List 3.1.6> Curved rule-like spline program for closed curve

```
/*  draft spline sample program for closed curve */
    set_blend(b, first_b, last_b, SPLIT);   /* set coeffcients */
    k = 0;                                  /* data counter */
    while( k < n-3 ) {
        mk_blend(&x1[k], &y1[k], wx, wy, b, SPLIT);
        mk_curve(wx, wy, SPLIT, c1, x0, y0, kx, ky, y_height);
        k++;
    }
    getch();
    closegraph();
}
```

•Execution example•